IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patent Application

Applicant(s):	Jai et al.
Docket No:	5-4-52
Serial No.:	10/600,995
Filing Date:	June 20, 2003
Group:	2446
Examiner:	Benjamin R. Bruckart
Title:	Automated Transformation of Specifications for Devices into Executable Modules

---

APPEAL BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

Applicants hereby appeal the final rejection dated October 28, 2008, of claims 1 through 21 of the above-identified patent application.

REAL PARTY IN INTEREST

The present application is assigned to Lucent Technologies Inc., as evidenced by an assignment recorded on October 27, 2003 in the United States Patent and Trademark Office at Reel 014642, Frame 0803. The assignee, Lucent Technologies Inc., is the real party in interest.

RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences.

STATUS OF CLAIMS

The present application was filed on June 20, 2003 with claims 1 through 21. Claims 1 through 21 are presently pending in the above-identified patent application. Claims 1-9, 14-16, 20, and 21 were rejected under 35 U.S.C. §102(e) as being anticipated over Moir

(United States Patent Publication Number 2002/120720), claims 10-14 were rejected under 35 U.S.C. §103(a) as being unpatentable over Moir in view of Tuatini (United States Patent Publication Number 2001/0047385), and claims 17-19 were rejected under 35 U.S.C. 103(a) as being unpatentable over Moir and in view of Presley (United States Patent Publication Number 2003/0105838).

Claims 1, 4, 8, 9, 14-17, 20, and 21 are being appealed.

STATUS OF AMENDMENTS

There have been no amendments filed subsequent to the final rejection.

SUMMARY OF CLAIMED SUBJECT MATTER

Independent claim 1 is directed to, in a system (FIG. 1: 100) having a plurality of devices (FIG. 1: 160), wherein a plurality of configuration elements are associated with the plurality of devices, a method for automated generation of executable modules associated with the devices (page 7, lines 5-29; page 10, lines 8-11; and FIG. 2), a method comprising the steps of:

accessing information about one or more input configuration elements of the plurality of configuration elements, wherein the one or more input configuration elements are associated with one or more input rules (FIG. 2: steps 210 and 220; page 7, lines 5-29; and page 10, lines 8-30);

determining which of the plurality of configuration elements could be accessed based on the one or more input rules (FIG. 2: steps 220 and 230; page 7, lines 5-29; and page 10, lines 20-30);

generating one or more output rules using at least the accessed information, the accessed configuration elements, and the input rules, wherein an output rule corresponds to one or more input configuration elements and wherein said one or more input rules comprise one or more executable statements (FIG. 2: step 240; page 10, line 31, to page 11, line 3); and

generating at least one executable module adapted to access at least a given one of the input configuration elements and to trigger one or more of the output rules corresponding to the given input configuration element (FIG. 2: step 250; page 20, lines 5-6).

Claim 4 requires wherein the step of determining read and write sets of configuration elements further comprises the step of determining for the given rule a call chain emanating from the given rule (page 17, line 12, to page 18, line 3).

Claim 8 requires wherein the step of determining which of the plurality of configuration elements could be accessed further comprise the step of determining, for a given one of one or more configuration elements able to be accessed by an input rule, a set of instance chain accesses for the given configuration element (page 16, line 21, to page 17, line 11).

Claim 9 requires wherein the given configuration element comprises a configuration element of a configuration class, wherein the given configuration element is another configuration class, and wherein the step of determining, for a given one of one or more configuration elements able to be accessed by an input rule, a set of instance chain accesses for the given configuration element further comprises the step of determining every access for the other configuration class to other configuration elements (page 21, lines 17-21).

Claim 14 requires wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through deferred triggering of the one or more output rules (page 16, lines 3-20; and page 19, lines 8-11).

Claim 15 requires wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through direct triggering of the one or more output rules (page 16, lines 3-20; and page 19, lines 5-7).

Claim 16 requires wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through batch triggering of the one or more output rules (page 16, lines 3-20; and page 19, lines 12-17).

Claim 17 requires wherein the one or more output rules comprise two or more output rules, and wherein the method further comprises the step of performing a circularity check by determining dependency relationships between the two or more output rules and by determining whether a given one of the two or more output rules depends upon itself (page 19, lines 18-22; and page 21, lines 26-30).

Independent claim 20 is directed to, in a system (FIG. 1: 100) having a plurality of devices (FIG. 1: 160), wherein a plurality of configuration elements are associated with the plurality of devices, an apparatus for automated generation of executable modules associated with the devices (page 7, lines 5-29; page 10, lines 8-11; and FIG. 2), an apparatus comprising:

a memory (FIG. 1: 107); and

at least one processor (FIG. 1: 106), coupled to the memory;

the apparatus being operative:

to access information about one or more input configuration elements of the plurality of configuration elements, wherein the one or more input configuration elements are associated with one or more input rules (FIG. 2: steps 210 and 220; page 7, lines 5-29; and page 10, lines 8-30);

to determine which of the plurality of configuration elements could be accessed based on the one or more input rules (FIG. 2: steps 220 and 230; page 7, lines 5-29; and page 10, lines 20-30);

to generate one or more output rules using at least the accessed information, the accessed configuration elements, and the input rules, wherein an output rule corresponds to one or more input configuration elements and wherein said one or more input rules comprise one or more executable statements (FIG. 2: step 240; page 10, line 31, to page 11, line 3); and

to generate at least one executable module adapted to access at least a given one of the input configuration elements and to trigger one or more of the output rules corresponding to the given input configuration element (FIG. 2: step 250; page 20, lines 5-6).

Independent claim 21 is directed to, an article of manufacture for use in a system (FIG. 1: 100) having a plurality of devices (FIG. 1: 160), wherein a plurality of configuration elements are associated with the plurality of devices, and for automated generation of executable modules associated with the device (page 7, lines 5-29; page 10, lines 8-11; and FIG. 2), an article of manufacture comprising:

a machine readable medium containing one or more programs which when executed implement the steps of:

accessing information about one or more input configuration elements of the plurality of configuration elements, wherein the one or more input configuration elements are associated with one or more input rules (FIG. 2: steps 210 and 220; page 7, lines 5-29; and page 10, lines 8-30);

determining which of the plurality of configuration elements could be accessed based on the one or more input rules (FIG. 2: steps 220 and 230; page 7, lines 5-29; and page 10, lines 20-30);

generating one or more output rules using at least the accessed information, the accessed configuration elements, and the input rules, wherein an output rule corresponds to one or more input configuration elements and wherein said one or more input rules comprise one or more executable statements (FIG. 2: step 240; page 10, line 31, to page 11, line 3); and

generating at least one executable module adapted to access at least a given one of the input configuration elements and to trigger one or more of the output rules corresponding to the given input configuration element (FIG. 2: step 250; page 20, lines 5-6).

STATEMENT OF GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1-9, 14-16, 20, and 21 were rejected under 35 U.S.C. §102(e) as being anticipated over Moir, claims 10-14 were rejected under 35 U.S.C. §103(a) as being unpatentable over Moir in view of Tuatini, and claims 17-19 were rejected under 35 U.S.C. 103(a) as being unpatentable over Moir and in view of Presley.

ARGUMENT

Independent Claims 1, 20 and 21

Independent claims 1, 20, and 21 were rejected under 35 U.S.C. §102(e) as being anticipated by Moir. Regarding claim 1, the Examiner asserts that Moir discloses generating one or more output rules using at least the accessed information, the accessed configuration elements, and the input rules, wherein an output rule corresponds to one or more input configuration elements and wherein said one or more input rules comprise one or more executable statements (page 5, paragraph 58)); and generating at least one executable module adapted to access at least a given one of the input configuration elements and to trigger one or more of the output rules corresponding to the given input configuration element (page 5, paragraph 60). In the Advisory Action, the Examiner asserts that the "configurations are executed to determine and control on how a device is to behave" (paragraph [0056]). The Examiner also asserts that "executable statements is broad and is not limited to code or a certain type of statement" and that "all the Moir reference has to show is that the statements are utilized or executed to perform an operation." The Examiner asserts that the rule program is derived by compiling the rule file and operations file (paragraph [0058]).

Appellants note that the present specification teaches that

> *input rules* are also part of specifications for a device and comprise, for example, *a set of checks or constraints or both* that should be performed before or after a configuration element is accessed. The input rules are generally derived from 'domain experts' (typically network specialists). An input rule is *usually represented as a set of executable statements*.
> (Page 2, lines 24-28.)

The present specification also teaches that

> *output rules* are *determined by using the accessed configuration elements, the input rules, and the way the input rule manipulates its accessed configuration elements*. Regarding the latter, output rules may be determined to deal with modifications to configuration elements, as explained in more detail below. In an illustrative embodiment, *each output rule is generally derived from exactly one input rule and corresponds to the same input configuration element associated with that input rule*. Output rules may be derived from multiple input rules, if desired.
> (Page 3, lines 7-14.)

Finally, the present disclosure teaches that

> an *executable module* is generated that is *adapted to access at least a given one of the input configuration elements and to trigger one or more of the output rules corresponding to the given input configuration element*.
> (Page 3, lines 15-17.)

In the text cited by the Examiner, however, Moir teaches:

> [0058] *The virtual machine compiler 60 utilizes the operations file 62 and the rule file 64 to compile a rule program 66*, which in one embodiment comprises a binary object including a sequence of instructions suitable for the virtual machine 10, discussed above. The rule program 66 comprises a set of operations, selected from operations supported by components of the network connection device 12, for performance by the respective components of the network connection device in accordance with the behavioral requirements defined by the rule file 64. In one embodiment, the rule program 66 may embody a number of sequences, these sequences constituting the classification rules 18, the event management rules 17 and the label management rules 19 discussed above with reference to FIG. 3.
> (Emphasis added.)

In the present Office Action, the Examiner asserts that paragraph 58 teaches that "the rule program is derived by compiling the rule file and operations file" (Office Action: page 3) and asserts that "the compiler binds processes behavior definitions and operations to data through compiling the operations file and rule files" (Office Action: page 10). Moir teaches, however, that the "*virtual machine compiler 60 utilizes the operations file 62 and the rule file 64 to compile a rule program 66*." (Emphasis added.) The Examiner has changed this statement to assert that Moir teaches "*compiling the operations file and rule files*." The statement "compiling a rule file" has a different meaning than "compiling a rule program." The Examiner's statement implies that the rule file contains executable statements; the teaching in Moir does *not* imply that the rule file contains executable statements. Moir does *not* teach that the rule file is compiled and does *not* disclose or suggest that *one or more input rules comprise one or more **executable statements***. Independent claims 1, 20, and 21 require *wherein said one or more input rules comprise one or more **executable statements***.

Regarding the Examiner's assertion that the "configurations are executed to determine and control on how a device is to behave" (paragraph [0056]), Appellants note that MOIR teaches:

> [0056] According to one embodiment of the present invention, a proposed solution to address the above identified network management problems includes *compiling the outcome of a number of discrete configuration steps* into an indivisible rule, which instructs a network device how to behave. This result may provide the advantage of allowing *configuration tasks* to be performed more reliably (and with a smaller code footprint), and also provides a mechanism for increasing the resolution of configuration without an adverse effect on the device's MTEF. Increased management resolution allows a network designer, for example, to safely exert control over very detailed aspects of the behavior of a network device, such as flow classification and data path features.
> (Emphasis added.)

Moir discloses *configuration steps*; Moir does not disclose or suggest, however, that *configuration files are executed.*

Regarding the Examiner's assertion that "executable statements is broad and is not limited to code or a certain type of statement" and that "all the Moir reference has to show is that the statements are utilized or executed to perform an operation," Appellants note that an "executable statement" is a statement that is executable, as would be apparent to a person of ordinary skill in the art. For example, MSDN (http://msdn.microsoft.com/en-us/library/59b7dyw0(VS.80).aspx) teaches that:

> *An executable statement performs an action.* It can call a procedure, branch to another place in the code, loop through several statements, or evaluate an expression. An assignment statement is a special case of an executable statement.
> (Emphasis added.)

Moir, alternatively, teaches:

> [0057] FIG. 9 is a block diagram providing a high level diagrammatic representation of the operation of a virtual machine compiler 60, according to an exemplary embodiment of the present invention. The virtual machine compiler 60 is shown to receive as inputs: (1) *an operations file 62 that describes operations supported by components of a particular network device (i.e., component behavior) and constraint definitions, and (2) a rule rile 64 that specifies behavioral requirements of a specific network device*. In one embodiment, these behavioral requirements may be specified as a textual representation in the form of a decision tree.
> (Emphasis added.)

Contrary to the Examiner's assertion, Moir does *not* teach that the rule file is compiled and does *not* disclose or suggest that *one or more input rules comprise one or more* ***executable statements***.

Thus, Moir, Tuatini and Presley, alone or in combination, do not disclose or suggest generating one or more output rules using at least the accessed information, the accessed configuration elements, and the input rules, wherein an output rule corresponds to one or more input configuration elements and wherein said one or more input rules comprise one or more executable statements; and generating at least one executable module adapted to access at least a given one of the input configuration elements and to trigger one or more of the output rules corresponding to the given input configuration element, as required by independent claims 1, 20, and 21.

<u>Claims 4 and 8</u>

Claims 4 and 8 were rejected under 35 U.S.C. §102(e) as being anticipated over Moir. Regarding claim 4, the Examiner asserts that Moir discloses wherein the step of determining read and write sets of configuration elements further comprises the step of determining for the given rule a call chain emanating from the given rule (Moir: page 8, continued from rule file page 7; the component may have multiple operations with the same identifier but with different argument types; calling different operations). Regarding claim 8, the Examiner asserts that Moir discloses wherein the step of determining which of the plurality of configuration elements could be accessed further comprise the step of determining, for a given one of one or more configuration elements able to be accessed by an input rule, a set of instance chain accesses for the given configuration element (Moir: page 8, continued from rule file page 7; the component may have multiple operations with the same identifier but with different argument types; calling different operations).

Appellants note that the present application teaches, for example, that "the call chain can include calls by the rule to other items, such as additional rules and utility methods." (Page 3, lines 26-28.) Contrary to the Examiner's assertion, the fact that "the component may have *multiple operations* with the same identifier but with different argument types" does *not* teach a "call chain," as defined in the context of the present invention, and does *not* teach *a <u>set of instance chain accesses</u> for the given configuration elemen*t. Moir teaches the grammar of the body of a rule, but does *not* disclose or suggest wherein the step of determining read and write sets of configuration elements further comprises the step of determining for the given rule *a call chain emanating from the given rule,* and does *not* disclose or suggest *a <u>set of instance chain accesses</u> for the given configuration elemen*t.

Thus, Moir, Tuatini and Presley, alone or in combination, do not disclose or suggest wherein the step of determining read and write sets of configuration elements further comprises the step of determining for the given rule a call chain emanating from the given rule, as required by claim 4, and do not disclose or suggest wherein the step of determining which of the plurality of configuration elements could be accessed further comprise the step of determining, for a given one of one or more configuration elements able to be accessed by an input rule, a set of instance chain accesses for the given configuration element, as required by claim 8.

<u>Claim 9</u>

Claim 9 was rejected under 35 U.S.C. §102(e) as being anticipated over Moir. Regarding claim 9, the Examiner asserts that Moir discloses wherein the given configuration element comprises a configuration element of a configuration class, wherein the given configuration element is another configuration class (Moir: page 6, paragraph 72; number of rules defined within a rule file), and wherein the step of determining, for a given one of one or more configuration elements able to be accessed by an input rule, a set of instance chain accesses for the given configuration element further comprises the step of determining every access for the other configuration class to other configuration elements (Moir: page 8, continued from rule file page 7; invocation of more than operation with different arguments).

Contrary to the Examiner's assertion, "the invocation of more than operation with different arguments" does *not* teach *determining <u>every access</u> for the other configuration class to* *other configuration element.* Moir teaches the grammar of the body of a rule, but does *not* disclose or suggest *determining* _every access_ *for the other configuration class to other configuration element.*

Thus, Moir, Tuatini and Presley, alone or in combination, do not disclose or suggest wherein the given configuration element comprises a configuration element of a configuration class, wherein the given configuration element is another configuration class, and wherein the step of determining, for a given one of one or more configuration elements able to be accessed by an input rule, a set of instance chain accesses for the given configuration element further comprises the step of determining every access for the other configuration class to other configuration elements, as required by claim 9.

<u>Claims 14 and 16</u>

Claims 14 and 16 were rejected under 35 U.S.C. §102(e) as being anticipated over Moir. Regarding claim 14, the Examiner asserts that Moir discloses wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through deferred triggering of the one or more output rules (Moir: page 8, <operation> is similar to the definition in the spec that "deferred rules are invoked as a sequence, typically at the end of a configuration session"). Regarding claim 16, the Examiner asserts that Moir discloses wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through batch triggering of the one or more output rules (Moir: page 8, continued from rule file page 7; batch of operations with the same identifier but with different argument types).

Moir teaches the grammar of the body of a rule. Contrary to the Examiner's assertion, the <operations> disclosed by Moir does *not* teach that "deferred rules are invoked as a sequence, typically at the end of a configuration session" and does *not* teach *triggering output rules corresponding to the given input configuration element through* <u>*deferred triggering*</u> *of the one or more output rules.* Similarly, the operations with the same identifier (but with different argument types) disclosed by Muir does *not* teach *triggering output rules corresponding to the given input configuration element through* <u>*batch triggering*</u> *of the one or more output rules.* Moir teaches the grammar of the body of a rule, but does *not* disclose or suggest wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through deferred triggering of the one or more output rules, and does *not* disclose or suggest wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through batch triggering of the one or more output rules.

Thus, Moir, Tuatini and Presley, alone or in combination, do not disclose or suggest wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through deferred triggering of the one or more output rules, as required by claim 14, and do not disclose or suggest wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through batch triggering of the one or more output rules, as required by claim 16.

Claim 15

Claim 15 was rejected under 35 U.S.C. §102(e) as being anticipated over Moir. Regarding claim 15, the Examiner asserts that Moir discloses wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through direct triggering of the one or more output rules (Moir: page 9, paragraph 105).

In the text cited by the Examiner, Moir teaches:

> [0105] At block 86, the rule program 66 is loaded into the network connection device 12, responsive to a user (or manager) request. For example, the rule program 66 may be loaded into the network connection device 12 from a remote location on demand from a user or manager.

Thus, Moir teaches that the rule program 66 may be loaded into the network connection device 12 from a remote location on demand from a user or manager; Moir does *not* teach, however, *to <u>trigger output rules</u> corresponding to the given input configuration element through <u>direct triggering</u> of the one or more output rules*.

Thus, Moir, Tuatini and Presley, alone or in combination, do not disclose or suggest wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through direct triggering of the one or more output rules, as required by claim 15.

Claim 17

Claim 17 was rejected under 35 U.S.C. 103(a) as being unpatentable over Moir and in view of Presley. Regarding claim 17, the Examiner acknowledges that Moir fails to teach performing a circularity check, but asserts that Presley teaches a method (that) further comprises the step of performing a circularity check by determining dependency relationships between the two or more output rules and by determining whether a given one of the two or more output rules depends upon itself (page 4, paragraph 47) in order to provide reliable and predictable performance (page 1, paragraph 5).

In the text cited by the Examiner, Presley teaches:

> [0047] The core services 51 obtain the site-specific resource and component configuration XML doclets 53-55 by invoking probing service layers (not shown). The core services 51 analyze, advise, and correct out-of-bound and invalid configuration parameters and identify cyclic relationships. A set of new XML doclets 58 are generated and fielded to the individual network components 33 for implementation. The core services 51 also exports management, advising and alert services 56. The core services 51 can be exported through a browser service 57, as implemented on the management console 31 (shown in FIG. 2).

Thus, Presley teaches identifying cyclic relationships; Presley, however, does *not* teach performing a circularity check by <u>*determining dependency relationships*</u> *between two or more output rules* and by *determining whether a given one of the two or more output rules* <u>*depends upon itself*</u>.

Thus, Moir, Tuatini and Presley, alone or in combination, do not disclose or suggest the step of performing a circularity check by determining dependency relationships between the two or more output rules and by determining whether a given one of the two or more output rules depends upon itself in order to provide reliable, as required by claim 17.

<u>Conclusion</u>

The rejections of the cited claims under sections 102 and 103 in view of Moir, Tuatini and Presley, alone or in any combination, are therefore believed to be improper and should be withdrawn. The remaining rejected dependent claims are believed allowable for at least the reasons identified above with respect to the independent claims.

The attention of the Examiner and the Appeal Board to this matter is appreciated.

Respectfully,

[illegible]

Date: May 7, 2009

Kevin M. Mason
Attorney for Applicant(s)
Reg. No. 36,597
Ryan, Mason & Lewis, LLP
1300 Post Road, Suite 205
Fairfield, CT 06824
(203) 255-6560

CLAIMS APPENDIX

1. In a system having a plurality of devices, wherein a plurality of configuration elements are associated with the plurality of devices, a method for automated generation of executable modules associated with the devices, the method comprising the steps of:

accessing information about one or more input configuration elements of the plurality of configuration elements, wherein the one or more input configuration elements are associated with one or more input rules;

determining which of the plurality of configuration elements could be accessed based on the one or more input rules;

generating one or more output rules using at least the accessed information, the accessed configuration elements, and the input rules, wherein an output rule corresponds to one or more input configuration elements and wherein said one or more input rules comprise one or more executable statements; and

generating at least one executable module adapted to access at least a given one of the input configuration elements and to trigger one or more of the output rules corresponding to the given input configuration element.

2. The method of claim 1, wherein the one or more input configuration elements are described by one or more configuration classes and wherein the one or more input rules are described by one or more rule files.

3. The method of claim 1, wherein the step of determining which of the plurality of configuration elements could be accessed further comprises the step of determining read and write sets of configuration elements for a given one of the one or more rules.

4. The method of claim 3, wherein the step of determining read and write sets of configuration elements further comprises the step of determining for the given rule a call chain emanating from the given rule.

5. The method of claim 4, wherein the step of determining for a given rule a call chain emanating from the rule further comprises the steps of determining whether the given rule accesses one or more items and determining whether one or more other configuration elements are accessed by the one or more items.

6. The method of claim 5, wherein the one or more items comprise one or more rules or one or more utility methods.

7. The method of claim 5, wherein the step of determining read and write sets of configuration elements further comprises the steps of determining whether the one or more items accesses one or more additional items and determining whether one or more additional configuration elements are accessed by the one or more additional items.

8. The method of claim 1, wherein the step of determining which of the plurality of configuration elements could be accessed further comprise the step of determining, for a given one of one or more configuration elements able to be accessed by an input rule, a set of instance chain accesses for the given configuration element.

9. The method of claim 8, wherein the given configuration element comprises a configuration element of a configuration class, wherein the given configuration element is another configuration class, and wherein the step of determining, for a given one of one or more configuration elements able to be accessed by an input rule, a set of instance chain accesses for the given configuration element further comprises the step of determining every access for the other configuration class to other configuration elements.

10. The method of claim 1, wherein the step of generating at least one executable module further comprises the step of generating at least one class for a given one of the one or more output rules, the at least one class defining the at least one executable module.

11. The method of claim 10, wherein the at least one class comprises one or more statements adapted to access at least one given configuration element that corresponds to the one

or more output rules.

12. The method of claim 10, wherein each of the at least one classes comprises one or more methods adapted to access the at least one given configuration element.

13. The method of claim 12, wherein the access comprises reading, writing, or modifying the at least one given configuration element.

14. The method of claim 1, wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through deferred triggering of the one or more output rules.

15. The method of claim 1, wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through direct triggering of the one or more output rules.

16. The method of claim 1, wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through batch triggering of the one or more output rules.

17. The method of claim 3, wherein the one or more output rules comprise two or more output rules, and wherein the method further comprises the step of performing a circularity check by determining dependency relationships between the two or more output rules and by determining whether a given one of the two or more output rules depends upon itself.

18. The method of claim 1, wherein the information further comprises at least one range restriction corresponding to the given input configuration element and wherein the at least one executable module is adapted to ensure that the at least one range restriction is met when the given configuration element accessed by the one or more triggered output rules is assigned a value.

19. The method of claim 1, wherein the information further comprises at least one referential integrity restriction corresponding to the given input configuration element and wherein the at least one executable module is further adapted to ensure that the at least one referential integrity restriction is met when the given configuration element is accessed by the one or more triggered output rules.

20. In a system having a plurality of devices, wherein a plurality of configuration elements are associated with the plurality of devices, an apparatus for automated generation of executable modules associated with the devices, the apparatus comprising:

a memory; and

at least one processor, coupled to the memory;

the apparatus being operative:

to access information about one or more input configuration elements of the plurality of configuration elements, wherein the one or more input configuration elements are associated with one or more input rules;

to determine which of the plurality of configuration elements could be accessed based on the one or more input rules;

to generate one or more output rules using at least the accessed information, the accessed configuration elements, and the input rules, wherein an output rule corresponds to one or more input configuration elements and wherein said one or more input rules comprise one or more executable statements; and

to generate at least one executable module adapted to access at least a given one of the input configuration elements and to trigger one or more of the output rules corresponding to the given input configuration element.

21. An article of manufacture for use in a system having a plurality of devices, wherein a plurality of configuration elements are associated with the plurality of devices, and for automated generation of executable modules associated with the device, the article of manufacture comprising:

a machine readable medium containing one or more programs which when executed implement the steps of:

accessing information about one or more input configuration elements of the plurality of configuration elements, wherein the one or more input configuration elements are associated with one or more input rules;

determining which of the plurality of configuration elements could be accessed based on the one or more input rules;

generating one or more output rules using at least the accessed information, the accessed configuration elements, and the input rules, wherein an output rule corresponds to one or more input configuration elements and wherein said one or more input rules comprise one or more executable statements; and

generating at least one executable module adapted to access at least a given one of the input configuration elements and to trigger one or more of the output rules corresponding to the given input configuration element.

EVIDENCE APPENDIX

There is no evidence submitted pursuant to § 1.130, 1.131, or 1.132 or entered by the Examiner and relied upon by appellant.

RELATED PROCEEDINGS APPENDIX

There are no known decisions rendered by a court or the Board in any proceeding identified pursuant to paragraph (c)(1)(ii) of 37 CFR 41.37.